



Not Netflix movie database

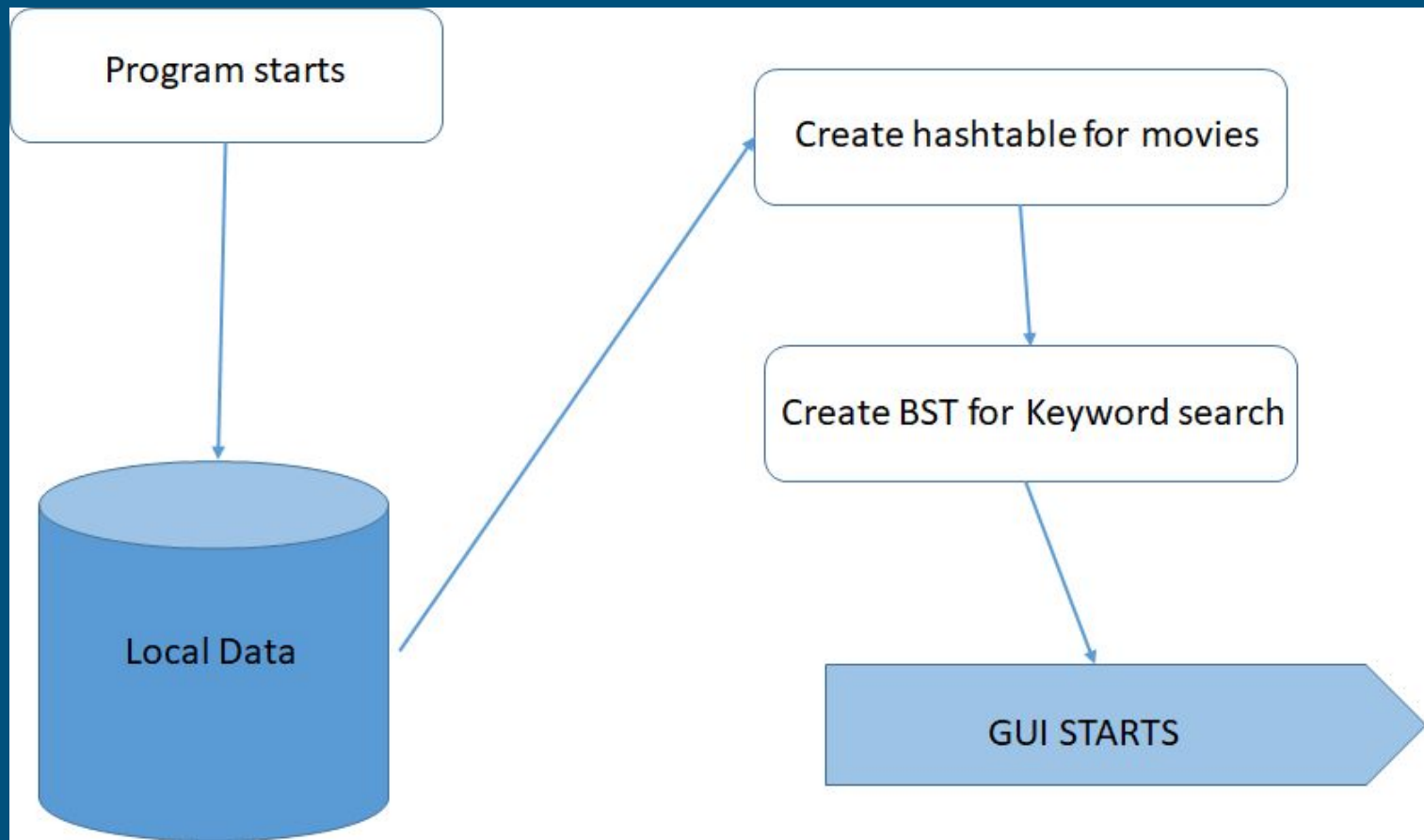


Henry, John, Ben



Overview

- Database to store information about movies
- Movie objects
 - Title
 - Year
 - Genre
 - Rating
 - ID
 - Runtime
- User can search with different keywords



Program workflow

Menu operations

- Search for a movie
 - View related movies
- Add a new movie
- Remove a movie
- Edit a movie
 - Pick from potential matches
- Undo a deletion
- Exit program

Data Structures

```
Welcome to the Movie Database

Enter the file path for an input file .tsv file
Example: C:\data\input.tsv
Hit enter to use the default .tsv file in this program's directory
```

```
title_basics_cleaned_final_trimmed_5000.tsv
```

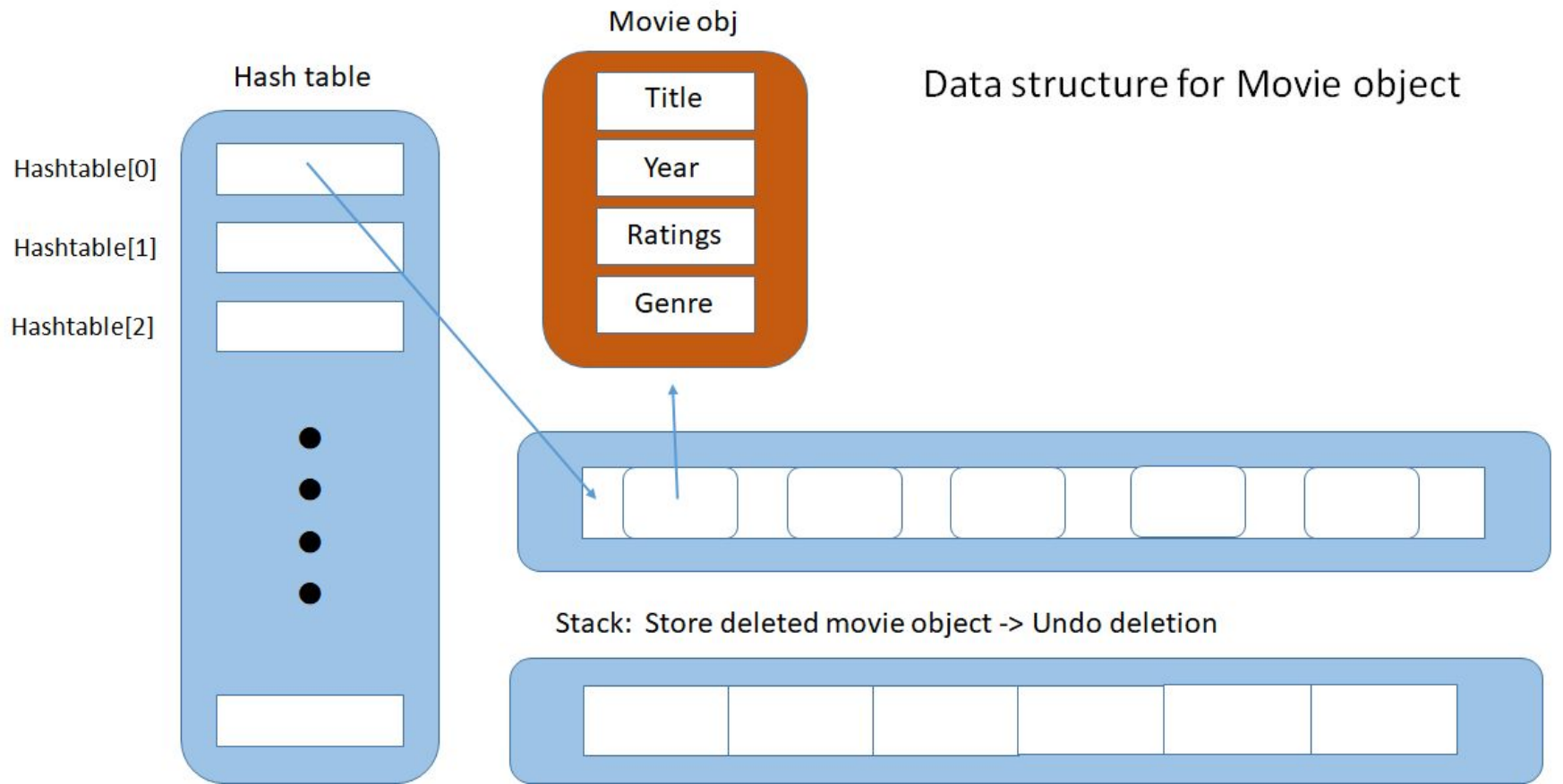
List of Movies Generation Time	0.607049s
Movie Load From List to Table Time	0.777294s
Tabled BST Search Engine Time	16.586s

Movie Table Statistics	
Occupancy	5052
Capacity	12637
Load Factor	39.98%
Number of Collisions	1021

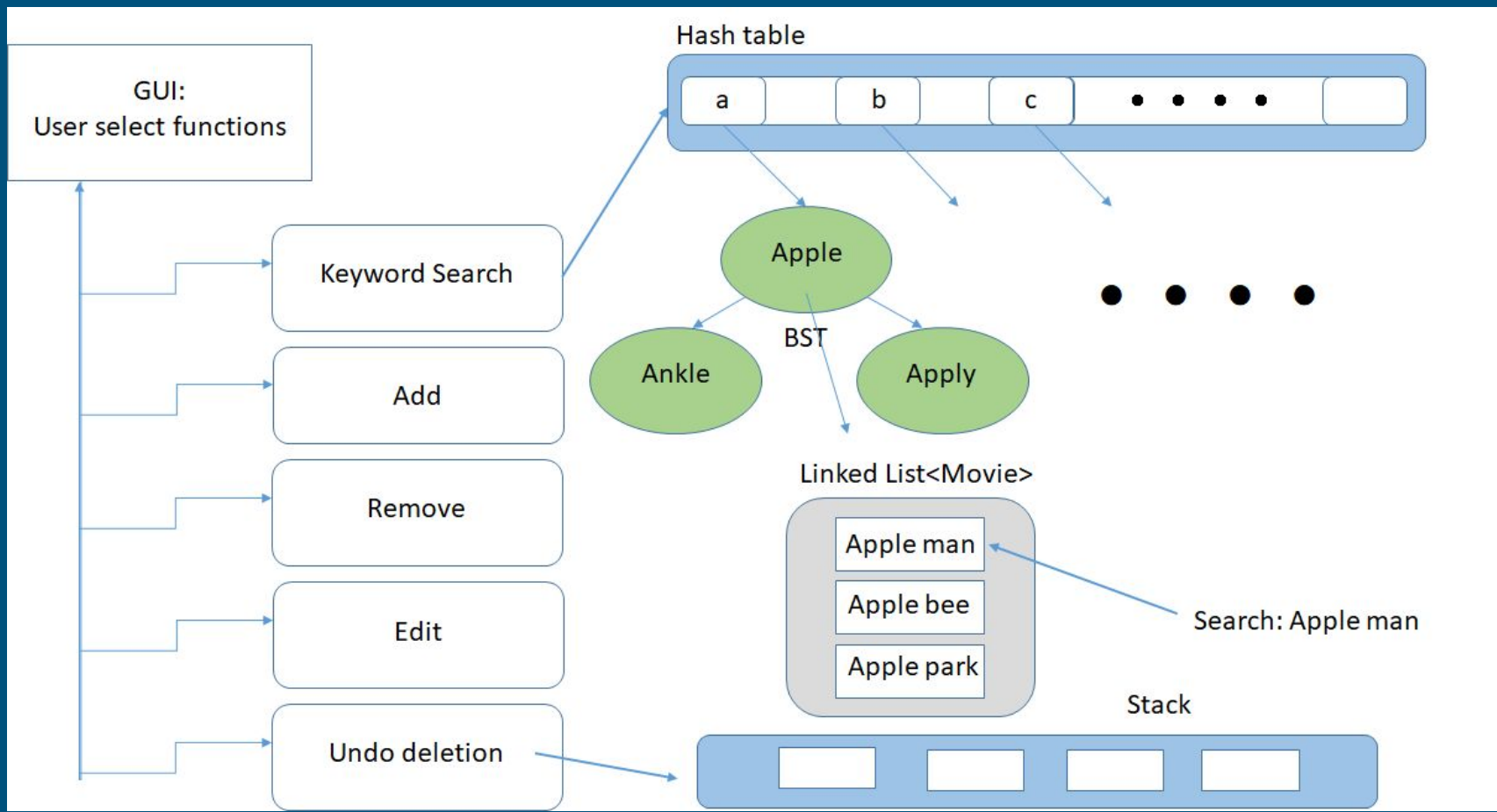
Tabled BST Search Engine Statistics	
Occupancy	43
Capacity	1009
Load Factor	4.262%
Number of Collisions	0

```
What would you like to do?
```

Performance



Structure diagram



Structure chart

Database Wrapper

```
HashTable<BinarySearchTree<string, Movie>*> *__searchEngineBST
```

```
HashTable<Movie>*> *__movieDB
```

```
Stack<Movie>*> *__deletedMovies
```

Binary Search Tree

```
template <typename K, typename V>
class BinaryNode {
private:
    K                __key;
    List<V>*         __values;
    BinaryNode*      __leftChild;
    BinaryNode*      __rightChild;
```

Binary Search Tree

- Adding a new tree

```
__searchEngineBST->add(std::string(1,  
    keywords->getEntry(j)[0]),  
    new BinarySearchTree<std::string,Movie*>(keywords->getEntry(j)));
```

- Updating the engine

```
(*__searchEngineBST)[firstCharOfKeyword]->addValue(keyword, editedMovie);
```

Hash Table

- Open addressing
 - Quadratic probing for collision resolution

```
h = (h << 4) + *sp;  
if (g = h & 0xF0000000) {  
    h ^= g >> 24;  
    h &= ~g;  
}
```

```
hashID = (hashID + size_t(0.5 * i) + size_t(0.25 * i * i)) % __capacity;
```

Hash Table

- Updating an attribute

```
std::string processedKey = processSearchEntry(key);  
Movie newMovie = __updateSearchEngineBST(  
    newYearReleased, (*__movieDB)[processedKey], 2);  
(*__movieDB)[processedKey] = new Movie(newMovie);
```

Linked List

- Deletion handler

```
std::string keyFound =  
    __searchEngineBST->get(firstCharOfKeyword)->getKey(keyword);  
List<Movie*>* keywordMovies =  
    __searchEngineBST->get(firstCharOfKeyword)->getValues(keyFound);  
keywordMovies->removeByValue(movieToDelete);
```

Demo



Q&A

